Continuous upstream

Page name could change when concept becomes clearer

Context

- As per Where to commit, new features generally shouldn't be committed to the stable branch. They belong in trunk.
  - You are running a Tiki, which is somewhat critical and/or has a lot of features and it takes quite a bit of time to test everything. You don't want down time and you need features to be stable. So you want to run a stable branch of Tiki (not trunk).
  - You are working on new features, which you want to contribute to trunk
  - You would like some of these new features as soon as possible available to your site and not wait up to 6 months for trunk to become the next stable version of Tiki.
  - You also have some non-upstreamable stuff (a theme and perhaps some things that are very specific to this project and not interesting for the community at large)
- You want to avoid custom code as much as possible and want each major upgrade to be as smooth as possible because all code/features has been upstreamed to trunk (Commit early, Commit often)

Examples

- Running a SaaS service, where releases of new features can be done every few weeks.

Workflow trunk to stable

Code in trunk and cherry-pick backports

Pros

- This is how the workflow *should* be.
- New code takes advantage of new trunk features
- Helping to keep trunk stable

Cons

- Trunk bugs (which are to be expected) can slow you down
- Sometimes, backport is messy because it involves something else only in trunk

Instances

- example.org: **live site**
  - stable branch
    - + own code already committed to trunk and backported locally
    - + own code that is not destined to be upstreamed (ex.: theme)
  - updated regularly to tip of stable branch
- **staging**.example.org
  - Generally same code and data as live site, but with a few recent changes that need to tested
- **next**.example.org: take production site and pre-dogfood server script (for code, not just data like current script does).

Important wish: TRIM make clone (mirror) and make cloneandupdate or cloneandupgrade (pre-dogfood server)

Steps

We will need to write scripts to automate this more, but here are steps for now.

1. Make sure your system requirements are sufficient to run trunk -> Server Check
2. Update your production code to the tip of the stable branch
    - Check if the update would cause conflicts
    - Check that nothing obvious is broken in the files that have been changed
    - You now have the tip of stable branch, along with your locally managed modifications on your site
3. Run pre-dogfood server script to get latest trunk and latest data on next
    - If this was run on a cron job, we could get an early warning with Check if the update would cause conflicts
4. Run script to make staging server be identical to production
5. Test the feature you are about to code on
    - If it's broken: How to figure out which commit causes a bug The sooner you do this, the less work it is.
6. Work on next. Make your feature.
    - If it involves modifying data (including prefs), you can use, see: System Configuration or Configuration Management and Systems Orchestration. You need this because your data will be wiped at the next pre-dogfood upgrade
7. Once you are pleased with it, merge all modified files to the staging server
8. Test. If all is good commit to trunk (in one commit, it makes it easier), and backport this commit to your stable site, along with any content / configuration changes
    - Your prod and staging should be identical at this point. A script to double-check this would be useful.

Workflow stable to trunk

Code on stable branch and merge to trunk

Pros

- Development on a more stable environment: fewer trunk bugs (which are to be expected) that can slow you down

Cons

- Not taking advantage of new trunk features
- Not helping much to keep trunk stable

Instances

- Similar to above

Steps

1. Make sure staging and next are up to date, in code and data
2. Code on staging
3. Test
4. Merge changes to next
5. Test
6. Commit to Tiki trunk
7. Backport this commit to live site
8. Run again scripts to update staging and next

Related

- Red Hat is a company with a policy we call “upstream first”
- https://trunkbaseddevelopment.com/

- Snapshots from trunk
- Dependency Injection
- http://12factor.net/dev-prod-parity
- Pre-Dogfood Server
- Configuration Management for Tiki Projects
- Divergent Preferences in Staging Development Production
- Continuous Integration
- Using Git with Tiki
- SUMO Upstream Process
- Translation upstream
- Semi-automatic merging period
- https://github.blog/2023-04-06-building-github-with-ruby-and-rails/
- Using GlitchTip as part of the Tiki development process